Development of an autonomous agent for the game **Snake**

Group Assignment

Inteligência Artificial

Ano Lectivo de 2024/2025

Diogo Gomes
Luís Seabra Lopes

October 9, 2024

I Important notes

1. This work must be carried out in groups of 2/3 students. In each Python module submitted, you must include a comment with the authors' name and mechanographic number.
2. A first version of the program must be submitted by 27th of November 2024. The final version must be submitted by 18th of December 2024. In both submissions, the work can be submitted beyond the deadline, but will be penalized in 5% for each additional day.
3. Each group must submit its code through the *GitHub Classroom* platform. In the final submission, include a presentation (Powerpoint type) in `.pdf` format and named `presentation.pdf`, with a maximum of five pages, where you should summarize the architecture of the developed agent.
4. The code should be developed in at least Python 3.11. The main module should be named `student.py`.
5. If you discuss this work with colleagues from other groups, include a comment with the name and mechanographic number of these colleagues. If you use other sources, cite them as well.
6. All code submitted must be original; though trusting that most groups will comply with this requirement, tools will be used to detect plagiarism. Students involved in plagiarism cases will have the assignment canceled.
7. The project will be evaluated taking into account: performance; quality of architecture and implementation; and originality.

II Overview

This work involves the application of concepts and techniques from three main chapters of the AI course, namely: Python programming; agent architectures; and search techniques for automated problem solving.

Within the scope of this work, you should develop an agent capable of playing intelligently the game Snake, as popularized by Nokia on its 6110 phone from 1998.

Based on the original Blockade game from 1976. Viewed from a top-down perspective, each player controls a "snake"with a fixed starting position. The "head"of the snake continually moves forward, unable to stop, growing ever longer. It must be steered to avoid hitting any obstacles, incuding the body of the opponent as well as its own body. It is usually played in multiplayer mode. The player who survives the longest wins. Single-player versions have one or more snakes controlled by the computer.

In this version of Snake we introduced some modifications. The main modification is that each snake has a limited view of the world, i.e. it does not see beyhond a given range from the head of the snake. Besides "normal"foods that grow the snake's body by one unit, there are super foods and poisons. Super foods can e.g. grow the snake more than one unit and/or increase sight of the snake. Poisons on the other hand can slim down the snake (removing units from the tail of the snake) or decrease its sight. Super Food and Poison are undistinguished, and the player only knows what it is when it eats it.

Super Foods and Poisons:

- POINTS: this can be either a super food (increases player score) or a poison (decrease players score) with random amount of points.

- LENGHT: this can either increase or decrease snake size (random amount [-5, 5])
- RANGE: this can either increase or decrease the snake's sight range (random amount [-5, 5])
- TRAVERSE: upon start the snake can run through walls and the world is a sphere, but when this poison is eaten the snake lives in a flat world and can crash into walls. Eating TRAVERSE a second time switches back the behavior.

Students must devise strategies to play in stand alone mode (scoring points for growing as much as possible within a time limit) and in multiplayer mode (same conditions but with extra snakes competing for the same resources).

1 Objectives

- To obtain a positive mark, the agent must be able to increase the size of the snake body by 100 times it's initial size.
- Score as much as possible. (see below details on marking)

III Game Rules

- *Snake* starts with a map which can either be empty or contain walls.
- The map is a sphere (wraps around itself) and the snake can crash against itself, other snakes (in the multiplayer version) and walls that might exist in the map. Walls can't be moved or created during the game. There is nonetheless a poison that can turn the borders of the map into walls (and the world becomes flat).
- The *Snake* player (student agent) has limited access to the map. The game server only passes map information in the vicinity of the head of the snake. Tthe range of view can change (increase or decrease) based on the food eaten.
- The agent controls a cursor, through which he can move the Snake either vertically or horizontally (diagonally is not allowed).
- In *Snake*, you can use the commands **"w"** (*move up*), **"s"** (*move down*), **"a"** (*move left*) and **"d"** (*move right*).
- A snake game is composed of a single map. Each game can have a different map.
- Each map in the game has a timeout value. If the snake doesn't kill itself, the game will terminate at the defined timeout value. In multiplayer games the game continues until all players have died or timeout.

IV Code and Development Support

A *Snake* game engine written in Python is available at `https://github.com/dgomes/ia-snake`.

All game entities are represented by classes.

Each group develops an agent creating a client that implements the protocol exemplified in the `client.py` file. No modifications to other files are necessary (you can't change `game.py`), but you can create new files, folders, etc.

If you implement a new feature or implement any improvement to the game engine and/or viewer, you can create a "Pull Request"(PR) on the GitHub platform. If your change is accepted, you will be credited with a bonus on the final evaluation up to a maximum of 1 point (in 20).

The developed agent must be delivered in a module named `student.py` and the agent should connect to the local game server, using as *username* the mechanographic number of one of the elements of the group (any).

There is a support channel in `https://detiuaveiro.slack.com/messages/ai/` where students can ask questions and receive notifications of changes.

Given the novelty of the game engine, it is expected that there are some bugs and tweaks during the course of work. Be on the watch out for server modifications (configure the professor repository as an upstream repository and fetch/merge often) and notifications in *e-mail*, *Slack* and *e-learning*.

To start the work, you must form a group with colleagues and access the link `https://classroom.github.com/a/rnI3I4bM` which will *fork* the code for the group. Only one *fork* should be done per group. One of the elements of the group creates the group's repository and associates the other elements. After this step, the *fork* will be created automatically (do not create a new *fork* without all elements being registered).

V Recommendations

1. Start by studying `client.py`. The code is very basic and simple, so start by *refactoring* the client to something more oriented to an autonomous AI Agent.
2. Periodically `git fetch` from the original repository to update your code.
3. Run `git log` to keep track of small changes that have been made.
4. Follow the channel #ai on Slack

VI Clarification of doubts

Clarifications on the main doubts that may arise during the performance of the work will be placed here.

1. **Question**: How will the performance of agents be evaluated?

   Answer: Agents will be evaluated on their performance in a set of games based on the sum of the final scores in these games. The final score in a game is the one the agent has at the time of *gameover*.

2. **Question**: How will the practical work be evaluated?

   Answer: Total game scores for each submitted agent are mapped to marks taking into acount the distribution of total scores. A total score of 100000 is mapped to 10/20. The median of the total scores is mapped to 16/20. The maximum total score is mapped to 20/20. Other total scores are mapped linearly.

   1st delivery evaluation

   - In this delivery, only the agent code must be submitted.
   - Each agent will play 10 games and the average of these 10 games will be obtained.

2nd delivery evaluation

- Each agent will play 10 games.
- In this delivery it is necessary to submit a presentation (see point I.3 above).
- The evaluation of the second delivery reflects the agent's performance (90%), the design quality (according to the delivered presentation, 7.5%) and the quality of the implementation (2.5%).